

Performance comparison of state-of-the-art deep learning model architectures in Indonesian food image classification

Mohammad Arif Rasyidi¹, Yunita Siti Mardhiyyah², Zuraidah Nasution³, Christofora Hanny Wijaya⁴

¹Department of Informatics, Universitas Internasional Semen Indonesia, Gresik, Indonesia

²Department of Agroindustrial Technology, Universitas Internasional Semen Indonesia, Gresik, Indonesia

³Department of Community Nutrition, IPB University, Bogor, Indonesia

⁴Department of Food Science and Technology, IPB University, Bogor, Indonesia

Article Info

Article history:

Received Jan 2, 2024

Revised Feb 11, 2024

Accepted Feb 28, 2024

Keywords:

Convolutional neural network

Dataset

Deep learning

Food recognition

Indonesian food

ABSTRACT

Food image recognition is essential for developing an elderly-friendly daily food recording application in Indonesia. However, existing datasets and models are limited and do not cover the diversity and complexity of Indonesian food. In this paper, we present a new dataset of 24,427 images of 160 types of Indonesian food with higher variety and quality than previous datasets. We also train and compare the performance of 67 models based on 16 state-of-the-art deep learning architectures on this dataset. We find that efficientnet_v2_l provides the best accuracy of 85.44%, followed by other models such as convnext_large and swin_s. We also discuss the trade-off between model size and performance, as well as the challenges and limitations of food image classification. Our dataset and models can serve as a basis for developing a user-friendly and accurate food recording application for the elderly population in Indonesia.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Yunita Siti Mardhiyyah

Department of Agroindustrial Technology, Universitas Internasional Semen Indonesia

PT Semen Indonesia (Persero) Tbk Complex, St. Veteran, Gresik, East Java, 6112, Indonesia

Email: yunita.mardhiyyah@uisi.ac.id

1. INTRODUCTION

According to the latest census data, Indonesia has a growing proportion of older people. In 2021, the percentage of people aged 60 and above was 10.8%, up from 8.9% in 2010 [1]. This number is expected to rise to one-fifth of Indonesia's population in 2045 [1]. The demographic shift poses significant challenges for the health and nutrition sector, as older people have different and often more complex needs than younger people. Some of the common health issues faced by older people in Indonesia are chronic diseases, such as diabetes, hypertension, and cardiovascular diseases; mental health problems, such as depression, dementia, and anxiety; disability and malnutrition. These issues require adequate and accessible healthcare services, social support, and education to improve their quality of life. However, many older people in Indonesia face barriers to accessing health care, such as lack of insurance, high costs, long distances, and stigma. Moreover, many older people lack awareness and knowledge about healthy aging and nutrition, which can lead to poor dietary habits and increased risk of diseases. Therefore, there is a need for more research, advocacy, and intervention to address the health and nutrition challenges of older people in Indonesia.

One critical factor affecting the life and health of older people is their daily dietary intake. By monitoring what and how much they eat, it is possible to obtain valuable information on their nutritional status, such as whether they are meeting their energy and nutrient requirements, whether they have any food intolerances or allergies, and whether they are at risk of developing malnutrition or chronic diseases such as diabetes, hypertension, or osteoporosis. Recording the daily food intake of older people can also help to

understand their dietary preferences, such as what foods they like or dislike, what foods they avoid or crave, and what foods they find difficult to chew or swallow. This can help caregivers and health professionals design appropriate interventions and recommendations to support their well-being, such as providing them with tailored menus, supplements, or counseling and educating them on the importance of a balanced and varied diet.

Although recording daily food intake is essential for monitoring and improving older people's health and well-being, conventional food recording methods, such as paper diaries, can pose many challenges and limitations for this population group. These methods require a lot of time and effort to record, measure, and estimate portion sizes, food types, and nutrient contents. They also rely on the memory and literacy skills of the users, which may decline with age. Furthermore, these methods may not capture the diversity and complexity of food consumption patterns and preferences among older people with different dietary needs, cultural backgrounds, and social contexts. Therefore, there is a need for more efficient, reliable, and user-friendly methods of food recording for the elderly population that can provide accurate and comprehensive information on their food intake and nutritional status.

One approach to overcoming the above problems is to use information technology. An elderly-friendly daily food recording application that can help improve their quality of life can be developed. Based on previous research, many daily food recording mobile applications have been developed [2], [3]. Some apps even have a feature to record food products based on user images [3]. Nevertheless, most of the current mobile apps for food logging do not include or cover traditional Indonesian food, which is a significant part of the diet of older people and the majority of Indonesians. This poses a challenge for accurately assessing the population's nutritional intake and health status, especially in rural areas where traditional foods are more common than other types of food.

Based on this background, one of our research objectives is to build a mobile application that can recognize traditional Indonesian food. We need to consider many factors in developing the application, such as usability, accessibility, user acceptance, and application functionality. In this study, we focused on one aspect of the application functionality, namely food recognition accuracy. A high-quality and diverse dataset of traditional Indonesian food is needed to develop an accurate food recognition model. However, this proved to be one of our biggest challenges. In previous research, [4], [5] have classified images of traditional Indonesian cakes using a deep learning approach. Unfortunately, their dataset only consists of cakes, so it does not cover most Indonesians' diets. Research by D. Sarwinda *et al.* [6] has also applied a convolutional neural network (CNN) to classify images of traditional Indonesian food. However, like [4], [5], even though the images they use do not only consist of cakes, they are limited to only ten types of food. Afrianto [7] also shared a dataset of 9 kinds of dishes from Padang, Indonesia, while [8] published a dataset of high-resolution images of 34 traditional Indonesian foods. Even though there are many types of food in the dataset shared by [8], the food images tend to be uniform and less varied because they were taken in a very controlled environment. So, if a prediction model is trained using only this data, it tends not to generalize well and will fail to predict various new images. What we need is data with many types of food consisting of diverse images. Thus, it is necessary to collect a comprehensive food image dataset to be used as a basis for developing prediction models for our application development needs.

The next challenge we face is choosing a suitable prediction method. In the last decade, deep learning algorithms, especially CNN and vision transformer (ViT), have become the de facto method for image classification. Deep learning has been applied in various fields and produces excellent performance. One factor determining deep learning's success in image classification is its architecture. Many deep learning architectures have been proposed and tested on the ImageNet dataset [9], a comprehensive image classification dataset consisting of 1000 classes, producing excellent performance, even exceeding human abilities in recognizing images. In previous research on food image recognition, deep learning has also been reported to produce good performance, for example, in [4], [5]. Unfortunately, only a few model architectures have been used and compared. Many new, recently developed model architectures have yet to be widely implemented and used. Thus, the question arises of how the latest architectures perform compared to older architectures that are more popular. Therefore, in this research, we propose a comparison of the performance of the latest deep learning architectures in classifying Indonesian food images. This comparison aims to find which architecture is best suitable for our dataset.

Our contribution is as follows. We present a novel Indonesian food image classification dataset encompassing a broader spectrum of food types, quantities, and variations compared to existing datasets. Additionally, we conduct a comprehensive comparative evaluation of the latest deep learning architectures for their performance on our newly compiled dataset. Our work is laying the groundwork for future personalized dietary assessment tools that leverage image recognition for accurate food intake monitoring as well as demonstrating the potential for scaling this approach to other diverse food cultures and applications beyond Indonesian cuisine.

The remainder of this paper will be organized as follows. Section 2 will discuss the methods we used in this research. The research results and discussion will be presented in section 3. Finally, conclusions and future work will be discussed in the last section.

2. METHOD

2.1. Data collection

As shown in Figure 1, this research began with the data collection stage. At this stage, we first determine the list of foods that will be used to develop the model in this research. With the abundance of Indonesian food and limited time and resources, not all food can be used in this research. Therefore, in this research, we chose 160 Indonesian foods as priorities for use. As shown in Table 1, these foods include 16 types of rice-based main dishes, 11 types of non-rice-based main dishes, 39 types of meat, chicken, egg, and fish-based side dishes, 17 types of vegetable-based side dishes, 23 types of soups, eight complementary foods, 19 traditional cakes, seven fruit and processed vegetables, ten drinks, and 11 snacks. Overall, the foods studied in this research have represented all the main islands in Indonesia, starting from Sumatra, Java, Kalimantan, Sulawesi, Bali, Nusa Tenggara, and Papua.

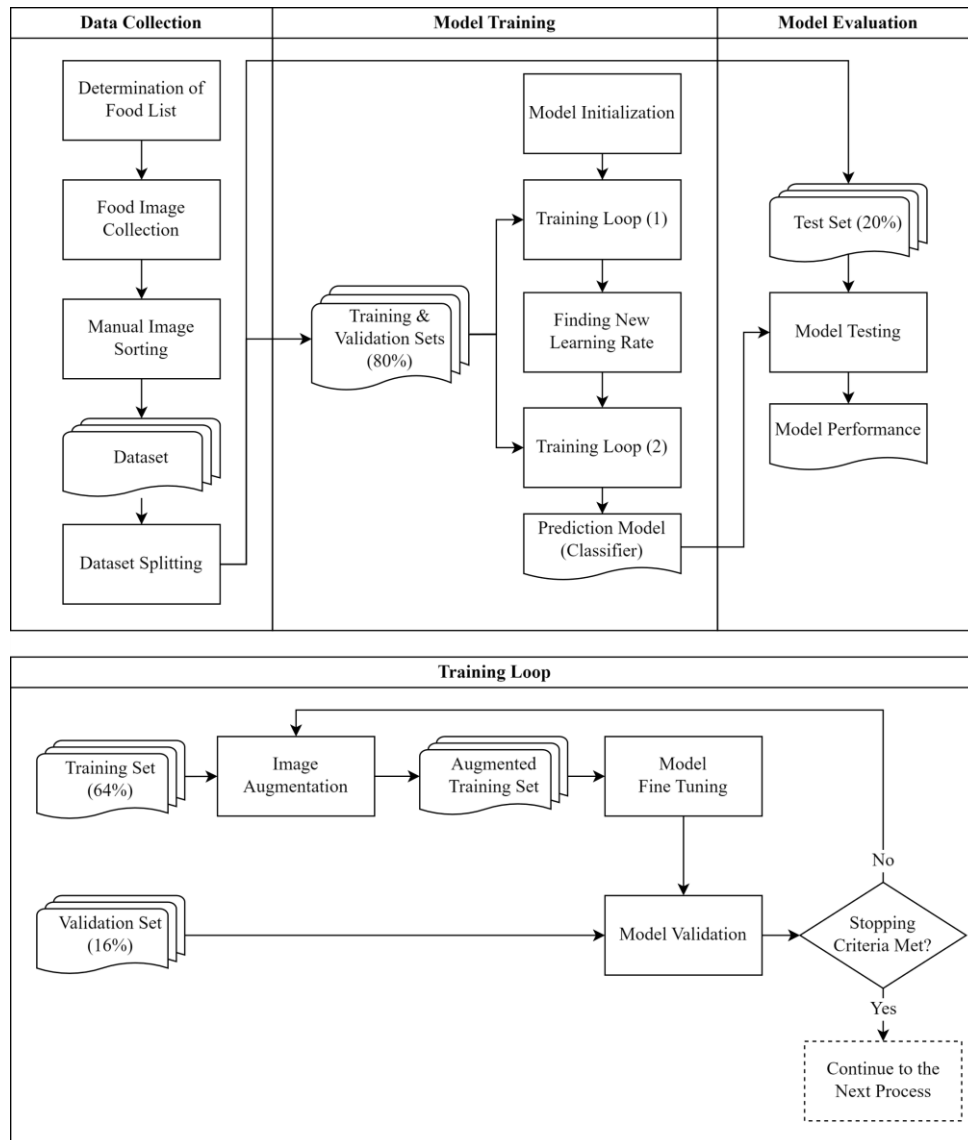


Figure 1. Research method

Once a list of foods was determined, we used an online search engine to collect images of those foods. The images obtained in this process are then divided and distributed to 10 human annotators in the manual sorting stage to label the images with the correct food class. Because the labeling is done manually by humans and one image is only handled by one person, there is the potential for errors in labeling food images. However, with the large number of images collected, this will not significantly affect the performance of the resulting model.

Table 1. The list of Indonesian foods used in this study

Category	Food items
Rice-based main dishes	<i>nasi kuning, nasi tim ayam, nasi gandul, gudeg, nasi timbel, nasi liwet, nasi goreng, nasi uduk, nasi padang, nasi pecel, nasi campur, and nasi kucing</i>
Non-rice-based main dishes	<i>mie kangkung, mi aceh, nasi jagung, papeda, arem-arem, leman, bakso, tau campur, karedok, mi kocok, bubur ayam, gado-gado, ketoprak, and kupat tahu</i>
Meat, chicken, egg, and fish-based side dishes	<i>sate lilit, sate udang, iga penyet, ayam popo, sate klatak, sate kerang, serudnneg daging, udang balado, dendeng balado, pallubasa, kepiting saus padang, ayam geprek, sate buntel, pecel lele, bebek goreng, babi guling, ikan bakar, rica-rica, ayam bakar, gulai ayam, kaledo, rendang daging, opor ayam, ayam pelalah, sate ayam, semur ayam, ayam betutu, cacalang fufu, ayam serundeng, babi panggang, sate kambing, ikan goreng, ayam penyet, semur daging, gulai kambing, rendang ayam, ayam goreng, tengkleng, and krengsengan daging</i>
Vegetable-based side dishes	<i>buntill, perkedel, pepes tahu, oncom, tempe goreng, tempe bacem, terong balado, botok, tempe kering, tumis kangkung, plecting kangkung, lawar, urap, lalapan, sambal, dabu-dabu, and bawang goreng</i>
Soups	<i>sayur asem, soto padang, soto ayam, soto lamongan, soto bandung, soto padang, sayu rlodeh, cap cai, rawon, soto ceker, tongseng, soto betawi, konro, soto medan, coto makasar, empal gentong, soto tangkar, soto babat, sop kambing, and sop saudara</i>
Complementary foods	<i>bubur ketan hitam, bubur kacang hijau, selat solo, tekwan, rujak cingur, kerak telur, tinutuan, and pempek</i>
Traditional cakes	<i>lapis legit, roti buaya, kue ape, wajik, roti canai, bakpia pathok, kue lapis, dadar gulung, kue cucur, kue putu, klepon, lempur, kue putu ayu, gethuk, kue nagasari, wingko, serabi, risoles, and klappertart</i>
Fruit and processed vegetables	<i>durian, petai, acar, rujak buah, asinan buah, asinan betawi, and kolak</i>
Drinks	<i>soda gembira, kopi terbalik, es teler, teh talua, sekoteng, wedang jahe, kopi tubruk, bajigur, and bandrek</i>
Snacks	<i>cilok, otak-otak, cireng, pisang goreng, tempe mendoan, singkong goreng, siomay, batagor, krupuk, kripik, and rengginang</i>

From this stage, 24,427 images were collected as the dataset used in this research. We divide the dataset into training and validation sets and test sets. The training and validation sets consist of 80% images from each food, totaling 19,475 images. While the remaining 4952 images (20%) will be used as the test set. Some examples of images in this dataset can be seen in Figure 2.

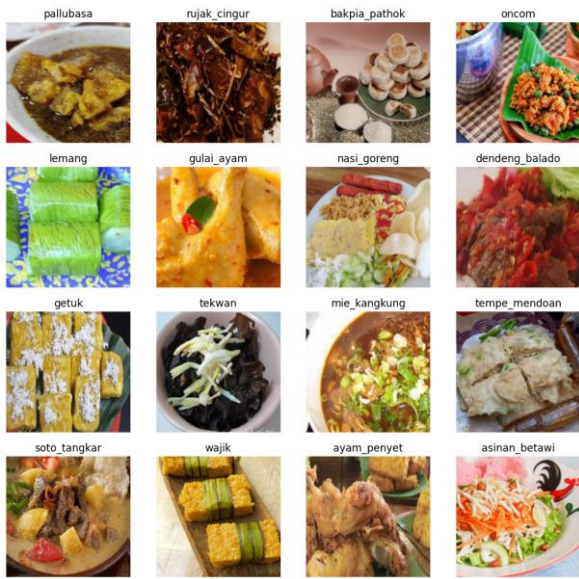


Figure 2. Some examples of food images in our dataset

2.2. Model training

We build image classification models at this stage using the latest deep learning architectures. We use 16 base deep learning architectures and their variants for a total of 67 model architectures, as shown in Table 2. We build one classification model for each model architecture. Model initialization is done by removing classification layers in the basic model and adding ones that suit our problem. Transfer learning is also carried out to speed up the training process. This is done by applying the weights of the model that has been previously trained using the ImageNet dataset to our model so that during training, we can focus on adjusting the weights in the classification layers and subsequent fine tunings.

Table 2. Deep learning architectures used in this study

Base architecture	Variants
AlexNet [10]	alexnet
ConvNeXt [11]	convnext_large, convnext_base, convnext_small, convnext_tiny
DenseNet [12]	densenet161, densenet201, densenet121, densenet169
EfficientNet [13]	efficientnet_b1, efficientnet_b2, efficientnet_b7, efficientnet_b0, efficientnet_b5, efficientnet_b6, efficientnet_b4, efficientnet_b3
EfficientNetV2 [14]	efficientnet_v2_l, efficientnet_v2_s, efficientnet_v2_m
GoogLeNet [15]	googlenet
MobileNetV3 [16]	mobilenet_v3_large, mobilenet_v3_small
RegNet [17]	regnet_y_3_2gf, regnet_y_16gf, regnet_x_32gf, regnet_x_16gf, regnet_x_8gf, regnet_y_32gf, regnet_y_8gf, regnet_x_3_2gf, regnet_y_1_6gf, regnet_y_800mf, regnet_x_800mf, regnet_x_1_6gf, regnet_y_400mf, regnet_x_400mf
ResNet [18]	resnet152, resnet101, resnet34, resnet50, resnet18
ResNeXt [19]	resnext101_64x4d, resnext101_32x8d, resnext50_32x4d
ShuffleNet v2 [20]	shufflenet_v2_x2_0, shufflenet_v2_x1_0, shufflenet_v2_x1_5, shufflenet_v2_x0_5
SqueezeNet [21]	squeezenet1_0, squeezenet1_1
Swin transformer [22]	swin_s, swin_b, swin_t
Swin transformer V2 [23]	swin_v2_b, swin_v2_s, swin_v2_t
VGG [24]	vgg16_bn, vgg19_bn, vgg13_bn, vgg11_bn, vgg19, vgg11, vgg16, vgg13
Wide ResNet [25]	wide_resnet50_2, wide_resnet101_2

The training and validation set obtained in the previous stage is then separated. We use the training set, which takes up 80% of these data or 64% of the initial data, to adjust the model weights during training. Meanwhile, the validation set is used to determine whether the training process still needs to be continued or can be stopped. The training process is carried out in two stages. In the first stage, using the training set, we trained the model for a maximum of 100 epochs using a 1cycle policy [26] and a base learning rate of 0.001.

During training, we use image augmentation to provide variations in the dataset used so that the model appears to see new data at each epoch. We use this to prevent overfitting so that the model can more accurately predict new data it has never seen before. The types of random transformations that we use in this augmentation process include rotation, mirroring, magnification, lighting, warping, and their combinations as shown in Table 3.

Table 3. Image augmentation parameters used in this study

Transformation	Parameter	Value	Probability
Flip	direction	horizontal	0.5
Rotation	max degree	10.0	0.75
Zoom	range	(1.0, 1.1)	0.75
Brightness and contrast change	max scale	0.2	0.75
Perspective warp	magnitude	0.2	0.75

At the end of each epoch, we validate the model against the validation set to obtain the validation loss. We apply early stopping to stop the training process if the validation loss at the end of an epoch does not improve in two consecutive epochs. After the first training stage ends, we look for a new learning rate by training the model for several iterations with different learning rates and recording the resulting loss. The learning rate that we use is at the midpoint of the longest valley in the learning rate vs loss curve. Next, we repeat the previous training procedure in the same way. Based on our experiments, this method consistently produces good results for our datasets. This training process is carried out using a system equipped with an NVIDIA GeForce RTX 4090 GPU and 32 GB RAM. During the training process, we also monitor training and validation loss at each epoch as well as training duration for further analysis.

2.3. Model testing

The models trained in the previous stage will be tested using our prepared test set. The test results for each model will be summarized using a confusion matrix, as shown in Figure 3. Based on the confusion matrix, accuracy as well as precision and recall for each type of food can be calculated using as (1) to (4):

$$accuracy = \frac{\sum_{i=1}^m TP_i}{n} \quad (1)$$

$$recall_i = \frac{TP_i}{TP_i + FN_i} \quad (2)$$

$$precision_i = \frac{TP_i}{TP_i + FP_i} \quad (3)$$

$$F1\ score_i = 2 \cdot \frac{precision_i \cdot recall_i}{precision_i + recall_i} \quad (4)$$

where TP_i is number of true positives for food i , FN_i is number of false negatives for food i , FP_i is number of false positives for food i , i is index of the food, m is number of classes, and n is number of data.

We also calculate top- k accuracy for $k=2, 3, 4$, and 5 by computing the number of times the correct label is among the top k predicted labels.

	Food_1	TP_1	FP_2/FN_1	FP_3/FN_1	FP_4/FN_1	...	FP_m/FN_1
	Food_2	FP_1/FN_2	TP_2	FP_3/FN_2	FP_4/FN_2	...	FP_m/FN_2
	Food_3	FP_1/FN_3	FP_2/FN_3	TP_3	FP_4/FN_3	...	FP_m/FN_3
Actual	Food_4	FP_1/FN_4	FP_2/FN_4	FP_3/FN_4	TP_4	...	FP_m/FN_4

	Food_m	FP_1/FN_m	FP_2/FN_m	FP_3/FN_m	FP_4/FN_m	...	TP_m
	Food_1	Food_2	Food_3	Food_4	...	Food_m	
	Prediction						

Figure 3. Sample confusion matrix

3. RESULTS AND DISCUSSION

3.1. Training results

Figure 4 illustrates the relationship between the number of model parameters (the size of the models) and the total epoch, time per epoch, and total training time. There is no clear correlation between the number of model parameters and the total number of epochs, meaning that models with more parameters do not necessarily train for more or fewer epochs. On the contrary, there is generally a positive correlation between the number of model parameters and the time per epoch, indicating that larger models tend to take longer to train for one epoch. On the other hand, although the correlation between model parameters and total training time is positive, it is not as strong as the correlation between model parameters and time per epoch. Hence, the total training time is not solely dependent on the number of model parameters. Other factors, such as the model architecture, may also play a role.

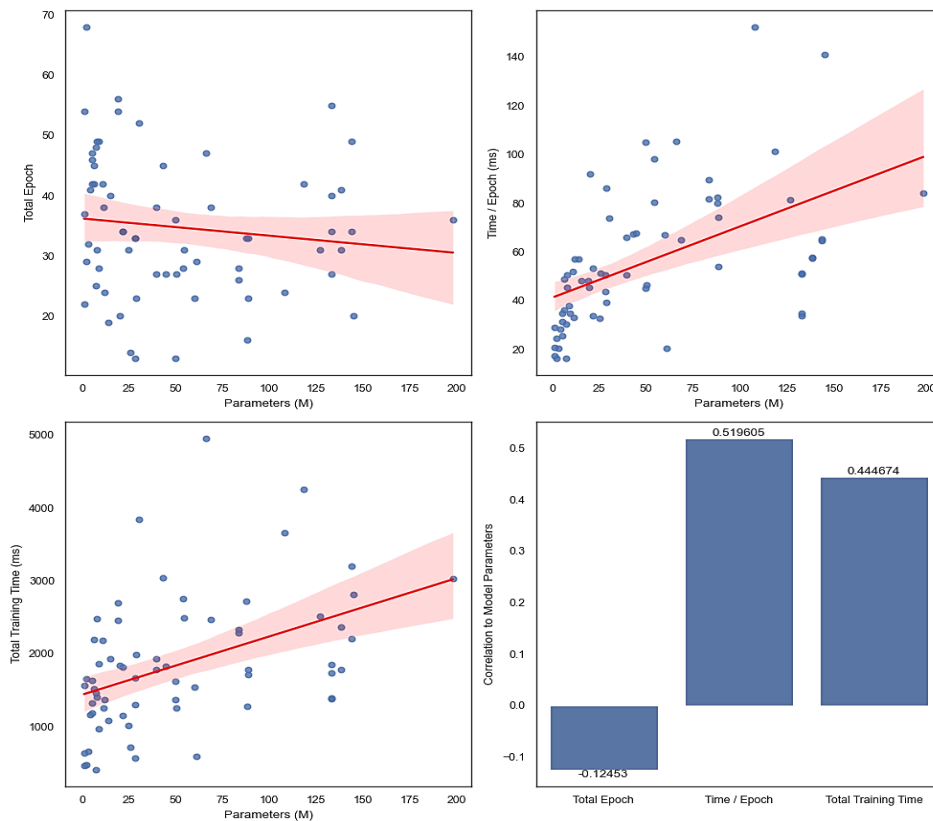


Figure 4. Relationship between the number of model parameters and total epoch, time per epoch, and total training time. Each dot represents a model. The red line is the trend line, while red shades show the 95% confidence interval

Figure 5 shows the training and validation loss curve for all models. As can be seen, training and validation loss successfully decreased, converged, and stabilized at the end of the training process for most of our models. However, some models, such as AlexNet, ShuffleNet v2, and swin transformer V2, failed to achieve this convergence, which indicates underfitting. Underfitting is a common issue in deep learning. In some cases, it can be a sign of insufficient data, noisy labels, or improper hyperparameters. In this study, we leave the investigation of underfitting for future works and focus on the models that have achieved satisfactory performance on the validation set.



Figure 5. Training vs validation loss

3.2. Overall model performance

Table 4 (in Appendix) shows the overall performance of our classification models. Most of the models show good performance in classifying Indonesian food images. The best performance was demonstrated by efficientnet_v2_l with an accuracy of 85.44% and a top-5 accuracy of 97.84%, but other models included in the top 10 also showed similar performance. Conversely, it is unsurprising that models that experience underfitting during the training process, such as AlexNet, ShuffleNet v2, and swin transformer V2, would demonstrate inferior performance compared to other models. This is particularly evident in the case of ShuffleNet v2, which exhibits very poor performance across all its variants. They can

only achieve accuracy between 9-13%, with top-5 accuracy between 27-34%. A plausible reason for this could be that the ShuffleNet v2 model variants, due to their small size and emphasis on efficiency, may not be sufficiently complex to tackle this classification problem, leading to underfitting.

When grouped and averaged based on the model architecture, as shown in Table 5, it can be seen that, generally, swin transformer provides the best performance for all of its variants. This fact is quite surprising because in contrast to other architectures, such as EfficientNetV2, where the latest version tends to have better performance than the old one, swin transformer far outperforms its newer variant, swin transformer V2, which is included among the three worst architectures. More recent model architectures such as swin transformer, ConvNeXt, and EfficientNetV2 also tend to provide better average performance than older models such as AlexNet, GoogleNet, and VGG. This is unsurprising, considering that newer models are usually developed to improve on old ones and are based on more advanced techniques such as transformers, depthwise convolution, and neural architecture search.

Table 5. Average performance of all classification models grouped by model architecture and ordered by best accuracy. The top five values are highlighted

Model architecture	Avg parameters (Million)	Avg accuracy	Avg top-2 accuracy	Avg top-3 accuracy	Avg top-4 accuracy	Avg top-5 accuracy	Avg accuracy/parameters
Swin transformer	55.2	0.8408	0.9291	0.9534	0.9663	0.9734	0.0186
ConvNeXt	91.3	0.8375	0.9255	0.9535	0.9664	0.9742	0.0148
EfficientNetV2	64.7	0.8201	0.9103	0.9402	0.9577	0.9669	0.0199
DenseNet	17.7	0.8174	0.9128	0.9421	0.9560	0.9648	0.0571
RegNet	39.2	0.8015	0.8962	0.9305	0.9477	0.9594	0.0626
ResNet	32.8	0.7963	0.8918	0.9261	0.9443	0.9554	0.0328
ResNeXt	65.8	0.7944	0.8926	0.9276	0.9451	0.9560	0.0167
Wide ResNet	97.9	0.7934	0.8896	0.9251	0.9444	0.9557	0.0089
VGG	137.0	0.7805	0.8818	0.9203	0.9398	0.9528	0.0057
EfficientNet	24.2	0.7726	0.8753	0.9146	0.9339	0.9475	0.0612
MobileNetV3	4.0	0.7613	0.8607	0.9035	0.9247	0.9386	0.2202
SqueezeNet	1.2	0.7295	0.8432	0.8907	0.9153	0.9300	0.6079
GoogLeNet	6.6	0.7292	0.8350	0.8841	0.9124	0.9311	0.1105
Swin transformer V2	55.3	0.6882	0.8085	0.8627	0.8907	0.9111	0.0150
AlexNet	61.1	0.6454	0.7767	0.8364	0.8740	0.8964	0.0106
ShuffleNet v2	3.7	0.1153	0.1850	0.2399	0.2810	0.3181	0.0430

3.3. Model size vs performance

As shown in Tables 4 and 5, the architecture that provides the best performance is usually not the most efficient, as indicated by the ratio between accuracy and the number of parameters. Figure 6 shows the same thing. There is a tendency for accuracy to increase with an increasing number of parameters in the model. However, this is only partially true when seen from each model's base architecture point of view. Figure 7 shows the relationship between the number of parameters and the performance of each architecture variant. For some architectures, such as EfficientNetV2 and ConvNeXt, their larger variants tend to perform better than their smaller ones as measured by accuracy. However, this relationship is not always true for other architectures, and even the opposite is true. For example, regnet_y_3_2gf has better accuracy than regnet_y_16gf and regnet_y_32gf while having a much smaller number of parameters. Meanwhile, efficientnet_b1, the second smallest variant of EfficientNet, has better accuracy than any other EfficientNet variants.

3.4. Which architecture to choose?

As explained in the previous section, a trade-off exists between model size and performance. Smaller models usually have lower accuracy, while larger models tend to perform better. On the other hand, smaller models are better suited for use in mobile devices because they require less memory and storage and can run more quickly and efficiently. Thus, we must balance this trade-off according to our needs and constraints. When requiring the highest accuracy without space, memory, and device limitations, efficientnet_v2_1 can be used. On the other hand, if we want a small model that can run on low-end devices and we are not too concerned about performance, squeezenet1_0, squeezenet1_1, and mobilenet_v3_small can be used.

One approach to selecting a suitable model is to define the threshold for each criterion that we require. For instance, in our scenario, we aim to choose a model that can operate on most devices while still achieving reasonably good accuracy. As depicted in Figure 8, we can establish the criteria that the model

must have an accuracy of at least 75%, and the number of parameters should not exceed 10 million. Then, any model that falls within the purple-shaded region, such as `densenet121`, `efficientnet_b1`, and `regnet_y_800mf`, can be selected.

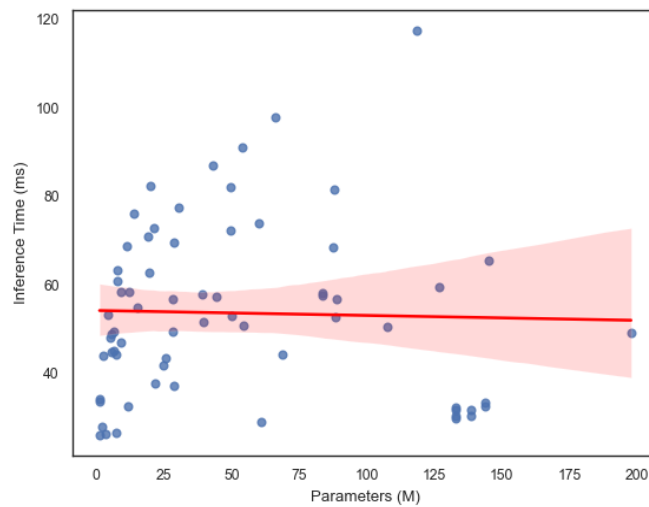


Figure 6. Relationship between the number of model parameters and accuracy. Each dot represents a model. The red line is the trend line, while red shades show the 95% confidence interval

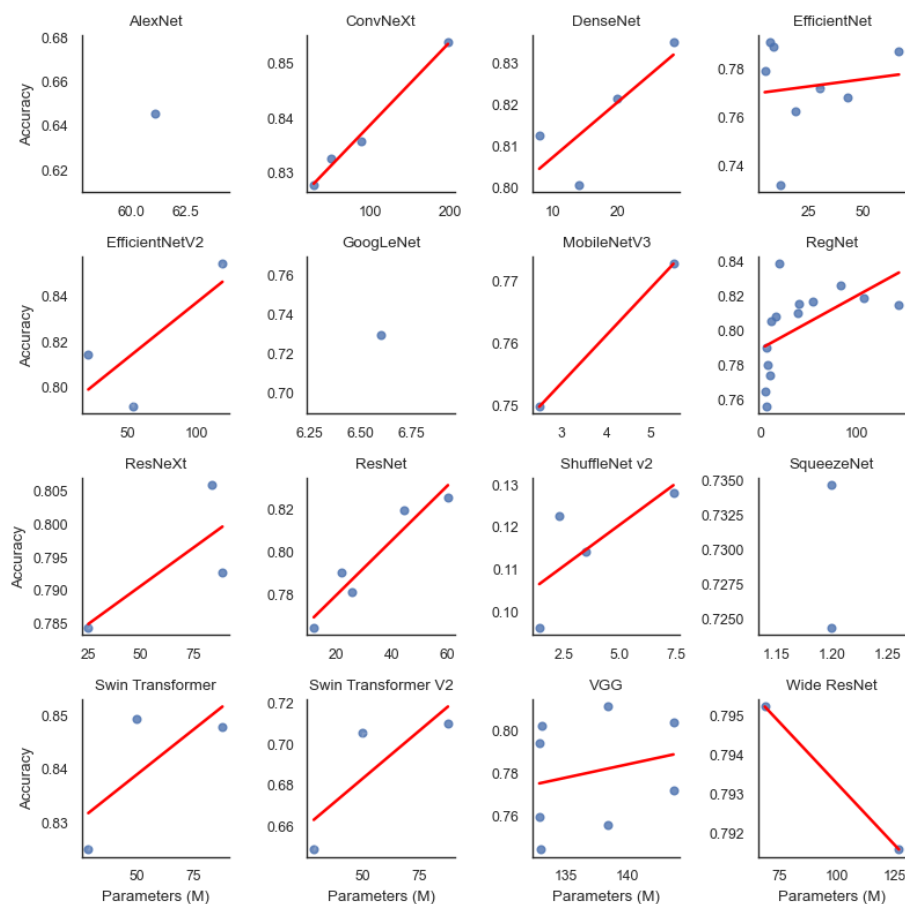


Figure 7. Relationship between the number of model parameters and accuracy grouped by model architecture. Each dot represents a model, while the red lines are the trend lines

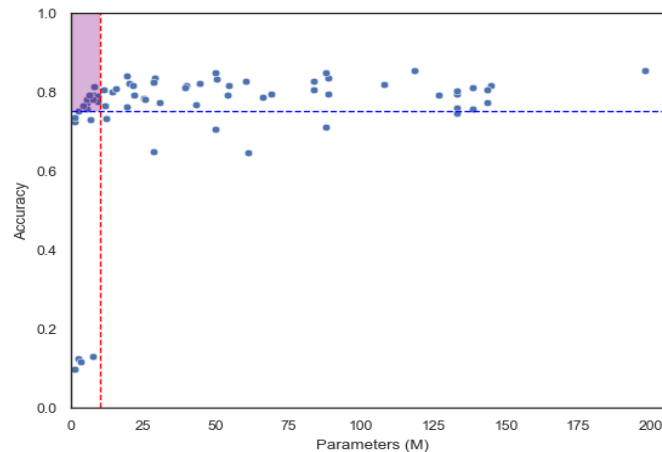


Figure 8. Illustration of model selection with an accuracy of at least 75% and no more than 10 million parameters. The purple-shaded area contains all suitable candidate models

3.5. Beyond model performance

Next, we analyze the test results of our best model, *efficientnet_v2_l*, to determine which types of food are most difficult for the model to predict. We use the results to evaluate our dataset and as input for future improvements. Table 6 shows the ten foods with the lowest F1 score. Our model generally has difficulty classifying *bandrek*, *sop saudara*, *kuluban*, and *sop kambing*, which are characterized by an F1 score below 0.5. One thing that might explain this is that in our dataset, *bandrek*, a hot drink made from ginger and brown sugar, and *sop saudara*, a type of beef soup, only have a few images compared to other foods. *Bandrek* consists of only 53 images, while *sop saudara* only consists of 67 images, far below the dataset average of 152 images for each food. Our model also often confuses *bandrek* with *bajigur*, a similar drink with different ingredients. With the similar appearance of the two, and coupled with the smaller number of *bandrek* datasets compared to *bajigur*, it is not surprising that our model has difficulty distinguishing between the two. A similar case occurred with *sop saudara*, *sop kambing*, and *kuluban*. The first two belong to a family of soups with thick broth, with their main ingredients partially submerged. This causes our model to often confuse them with similar foods such as *coto Makassar* and *konro*. On the other hand, *kuluban* is also often confused with *urap*, a food with similar ingredients and appearance.

Table 6. The performance of *efficientnet_v2_l* model on ten foods with the worst F1 score

Food item	F1 score	Precision	Recall
<i>Bandrek</i>	0.3158	0.3750	0.2727
<i>Sop saudara</i>	0.4211	0.6667	0.3077
<i>Kuluban</i>	0.4211	0.4444	0.4000
<i>Sop kambing</i>	0.4324	0.6154	0.3333
<i>Soto babat</i>	0.5185	0.5385	0.5000
<i>Krengsengan daging</i>	0.5313	0.5484	0.5152
<i>Gulai kambing</i>	0.5672	0.5758	0.5588
<i>Soto tangkar</i>	0.5789	0.7333	0.4783
<i>Tengkleng</i>	0.5926	0.6667	0.5333
<i>Ayam goreng</i>	0.6061	0.6250	0.5882

Looking at Figure 9, we find other challenges that our model faces in classifying food images. First, the variations in food presentation. One of *sate lilit*'s main characteristics is using lemongrass as a skewer. The first image in Figure 9 shows *sate udang*, which uses lemongrass instead of bamboo as the skewer. This deceives our model into predicting the image as *sate lilit*.

Furthermore, our image sometimes consists of several food items, as shown by the second image in Figure 9. Our model predicts the image as *nasi campur*, which is correct because it consists of rice and various side dishes. On the other hand, our data verifier labeled the image as *sate lilit* in our dataset, which is also not entirely wrong because one of the components in the image is *sate lilit*. Here, we see the limitations of our model in making predictions. Our classification model can only provide one prediction result for each image. Instead of classification, the more suitable tasks for images containing several labels or objects are multilabel classification, object detection, or semantic segmentation. We will explore this in future research.



Figure 9. Three most confused test images by efficientnet_v2_l model

4. CONCLUSION

A good food image prediction model is crucial for developing an elderly-friendly daily food recording application in Indonesia. In this research, we introduce a novel dataset encompassing 24,427 images across 160 unique Indonesian food types, surpassing existing datasets in both variety and image quantity. Through extensive benchmarking, we evaluated the performance of 67 models built upon 16 state-of-the-art deep learning architectures on our curated dataset. Notably, efficientnet_v2_l emerged as the top performer, achieving an accuracy of 85.44% and a top-5 accuracy of 97.84%, followed closely by other models such as convnext_large and swin_s. While newer models often result in better performance due to the use of more advanced techniques, they are not always better, as shown by swin transformer, which outperforms its successor, swin transformer V2. Our findings highlight the trade-off between model size and performance, where smaller models tend to exhibit lower accuracy while larger ones generally demonstrate better performance. However, this relationship is not universally applicable across all architectures, as demonstrated by certain smaller models outperforming their larger counterparts. Additionally, we have discussed considerations for model selection, acknowledging challenges and limitations surrounding food image classification. These limitations include the quantity and variability of images, food presentation styles, image composition complexities, and constraints inherent to single-label classification models. In future work, we will focus on expanding our dataset to encompass a wider range of Indonesian foods. To ensure label quality, we will implement multi-annotator labelling and explore techniques like confident learning. Furthermore, we plan to investigate the application of object detection techniques within this domain.

APPENDIX

Table 4. Performance of all classification models ordered by best accuracy. The top ten values are highlighted

Model architecture	Variant	Parameters (Million)	Accuracy	Top-2 Accuracy	Top-3 Accuracy	Top-4 Accuracy	Top-5 Accuracy	Accuracy/Parameters
EfficientNet V2	efficientnet_v2_l	118.5	0.8544	0.9317	0.9566	0.9719	0.9784	0.0072
ConvNeXt	convnext_large	197.8	0.8538	0.9370	0.9606	0.9719	0.9774	0.0043
Swin transformer	swin_s	49.6	0.8496	0.9319	0.9556	0.9675	0.9731	0.0171
Swin transformer	swin_b	87.8	0.8479	0.9344	0.9554	0.9707	0.9764	0.0097
RegNet	regnet_y_3_2gf	19.4	0.8389	0.9184	0.9465	0.9643	0.9733	0.0432
ConvNeXt	convnext_base	88.6	0.8358	0.9241	0.9529	0.9639	0.9733	0.0094
DenseNet	densenet161	28.7	0.8350	0.9255	0.9513	0.9639	0.9711	0.0291
ConvNeXt	convnext_small	50.2	0.8326	0.9227	0.9507	0.9659	0.9733	0.0166
ConvNeXt	convnext_tiny	28.6	0.8277	0.9184	0.9495	0.9639	0.9725	0.0289
RegNet	regnet_y_16gf	83.6	0.8265	0.9107	0.9435	0.9576	0.9669	0.0099
ResNet	resnet152	60.2	0.8257	0.9116	0.9426	0.9584	0.9665	0.0137
Swin transformer	swin_t	28.3	0.8249	0.9208	0.9493	0.9608	0.9707	0.0291
DenseNet	densenet201	20	0.8215	0.9152	0.9424	0.9544	0.9622	0.0411
ResNet	resnet101	44.5	0.8201	0.9097	0.9378	0.9546	0.9643	0.0184
RegNet	regnet_x_32gf	107.8	0.8189	0.9055	0.9382	0.9554	0.9659	0.0076
RegNet	regnet_x_16gf	54.3	0.8168	0.9073	0.9392	0.9542	0.9624	0.0150
RegNet	regnet_x_8gf	39.6	0.8158	0.9067	0.9370	0.9527	0.9620	0.0206
RegNet	regnet_y_32gf	145	0.8152	0.9073	0.9390	0.9558	0.9647	0.0056
EfficientNet V2	efficientnet_v2_s	21.5	0.8146	0.9095	0.9384	0.9568	0.9663	0.0379

Table 4. Performance of all classification models ordered by best accuracy. The top ten values are highlighted (*continue*)

Model architecture	Variant	Parameters (Million)	Accuracy	Top-2 Accuracy	Top-3 Accuracy	Top-4 Accuracy	Top-5 Accuracy	Accuracy/Parameters
DenseNet	densenet121	8	0.8126	0.9111	0.9424	0.9574	0.9669	0.1016
VGG	vgg16_bn	138.4	0.8112	0.9039	0.9376	0.9517	0.9616	0.0059
RegNet	regnet_y_8gf	39.4	0.8106	0.9069	0.9400	0.9554	0.9647	0.0206
RegNet	regnet_x_3_2gf	15.3	0.8080	0.9025	0.9332	0.9479	0.9610	0.0528
ResNeXt	resnext101_64x4d	83.5	0.8059	0.8992	0.9338	0.9513	0.9594	0.0097
RegNet	regnet_y_1_6gf	11.2	0.8055	0.9004	0.9328	0.9493	0.9610	0.0719
VGG	vgg19_bn	143.7	0.8039	0.8970	0.9346	0.9511	0.9639	0.0056
VGG	vgg13_bn	133.1	0.8025	0.9023	0.9340	0.9527	0.9616	0.0060
DenseNet	densenet169	14.1	0.8005	0.8992	0.9324	0.9485	0.9588	0.0568
Wide ResNet	wide_resnet50_2	68.9	0.7952	0.8918	0.9281	0.9461	0.9582	0.0115
VGG	vgg11_bn	132.9	0.7940	0.8901	0.9269	0.9439	0.9544	0.0060
ResNeXt	resnext101_32x8d	88.8	0.7928	0.8936	0.9289	0.9437	0.9574	0.0089
Wide ResNet	wide_resnet101_2	126.9	0.7916	0.8875	0.9221	0.9426	0.9532	0.0062
EfficientNet	efficientnet_v2_m	54.1	0.7914	0.8897	0.9257	0.9445	0.9562	0.0146
V2								
EfficientNet	efficientnet_b1	7.8	0.7908	0.8877	0.9253	0.9404	0.9546	0.1014
ResNet	resnet34	21.8	0.7904	0.8855	0.9237	0.9418	0.9548	0.0363
RegNet	regnet_y_800mf	6.4	0.7902	0.8910	0.9279	0.9443	0.9560	0.1235
EfficientNet	efficientnet_b2	9.1	0.7890	0.8859	0.9255	0.9429	0.9550	0.0867
EfficientNet	efficientnet_b7	66.3	0.7870	0.8873	0.9212	0.9390	0.9534	0.0119
ResNeXt	resnext50_32x4d	25	0.7843	0.8851	0.9202	0.9404	0.9513	0.0314
ResNet	resnet50	25.6	0.7813	0.8851	0.9178	0.9366	0.9495	0.0305
RegNet	regnet_x_800mf	7.3	0.7803	0.8849	0.9229	0.9404	0.9568	0.1069
EfficientNet	efficientnet_b0	5.3	0.7793	0.8813	0.9176	0.9368	0.9483	0.1470
RegNet	regnet_x_1_6gf	9.2	0.7744	0.8744	0.9134	0.9330	0.9489	0.0842
MobileNetV3	mobilenet_v3_large	5.5	0.7728	0.8724	0.9130	0.9334	0.9453	0.1405
EfficientNet	efficientnet_b5	30.4	0.7722	0.8762	0.9158	0.9392	0.9501	0.0254
VGG	vgg19	143.7	0.7722	0.8798	0.9144	0.9354	0.9511	0.0054
EfficientNet	efficientnet_b6	43	0.7680	0.8758	0.9103	0.9277	0.9433	0.0179
RegNet	regnet_y_400mf	4.3	0.7645	0.8667	0.9087	0.9295	0.9453	0.1778
ResNet	resnet18	11.7	0.7641	0.8671	0.9083	0.9301	0.9420	0.0653
EfficientNet	efficientnet_b4	19.3	0.7623	0.8621	0.9047	0.9265	0.9433	0.0395
VGG	vgg11	132.9	0.7597	0.8601	0.9039	0.9273	0.9431	0.0057
VGG	vgg16	138.4	0.7561	0.8673	0.9103	0.9321	0.9449	0.0055
RegNet	regnet_x_400mf	5.5	0.7559	0.8639	0.9045	0.9285	0.9433	0.1374
MobileNetV3	mobilenet_v3_small	2.5	0.7498	0.8489	0.8940	0.9160	0.9319	0.2999
VGG	vgg13	133	0.7445	0.8538	0.9006	0.9243	0.9422	0.0056
SqueezeNet	squeezenet1_0	1.2	0.7347	0.8439	0.8916	0.9166	0.9328	0.6122
EfficientNet	efficientnet_b3	12.2	0.7320	0.8459	0.8966	0.9190	0.9319	0.0600
GoogLeNet	googlenet	6.6	0.7292	0.8350	0.8841	0.9124	0.9311	0.1105
SqueezeNet	squeezenet1_1	1.2	0.7244	0.8425	0.8897	0.9140	0.9273	0.6036
Swin	swin_v2_b	87.9	0.7104	0.8277	0.8805	0.9097	0.9273	0.0081
transformer								
V2								
Swin	swin_v2_s	49.7	0.7056	0.8251	0.8790	0.9029	0.9223	0.0142
transformer								
V2								
swin	swin_v2_t	28.4	0.6486	0.7726	0.8286	0.8595	0.8839	0.0228
transformer								
v2								
AlexNet	alexnet	61.1	0.6454	0.7767	0.8364	0.8740	0.8964	0.0106
ShuffleNet v2	shufflenet_v2_x2_0	7.4	0.1282	0.2021	0.2559	0.3003	0.3407	0.0173
ShuffleNet v2	shufflenet_v2_x1_0	2.3	0.1228	0.1961	0.2561	0.2962	0.3330	0.0534
ShuffleNet v2	shufflenet_v2_x1_5	3.5	0.1141	0.1866	0.2399	0.2843	0.3233	0.0326
ShuffleNet v2	shufflenet_v2_x0_5	1.4	0.0961	0.1551	0.2076	0.2431	0.2752	0.0687





ACKNOWLEDGEMENTS

The authors thank the Directorate of General Higher Education, Research and Technology, Ministry of Education, Culture, Research and Technology of the Republic of Indonesia for research funding through the National Competitive Research on National Collaborative Research scheme in 2023.





REFERENCES

- [1] A. P. L. Girsang, K. D. Ramadani, S. W. Nugroho, N. P. Sulistyowati, R. Putrianti, and H. Wilson, "2021 Senior Population Statistics (in Bahasa: Statistik Penduduk Lanjut Usia 2021)," *Badan Pusat Statistik*, 2021, [Online]. Available: <https://www.bps.go.id/publication/2021/12/21/c3fd9f27372f6ddcf7462006/statistik-penduduk-lanjut-usia-2021.html>.
- [2] A. Myers *et al.*, "Im2Calories: Towards an automated mobile vision food diary," in *Proceedings of the IEEE International Conference on Computer Vision, International Conference on Computer Vision ICCV*, vol. 2015, pp. 1233–1241, 2015, doi: 10.1109/ICCV.2015.146.
- [3] S. van Asbroeck and C. Matthys, "Use of different food image recognition platforms in dietary assessment: Comparison study," *JMIR Formative Research*, vol. 4, no. 12, p. e15602, Dec. 2020, doi: 10.2196/15602.
- [4] T. Karlita, B. P. Afif, and I. Prasetyaningrum, "Indonesian Traditional Cake Classification Using Convolutional Neural Networks," in *Proceedings of the International Conference on Applied Science and Technology on Social Science 2021 (iCAST-SS 2021)*, vol. 647, 2022, doi: 10.2991/assehr.k.220301.153.
- [5] F. Aziz, "Recognition of Indonesian Traditional Cakes using The MobileNet Algorithm," *International Journal of Computer and Information Technology (2279-0764)*, vol. 11, no. 5, Jan. 2023, doi: 10.24203/ijcit.v11i5.254.
- [6] D. Sarwinda *et al.*, "Automatic Multi-class Classification of Indonesian Traditional Food using Convolutional Neural Networks," in *2020 3rd International Conference on Computer and Informatics Engineering, IC2IE*, Sep. 2020, pp. 43–47, 2020, doi: 10.1109/IC2IE50715.2020.9274636.
- [7] F. F. Afrianto, "Padang Cuisine (Indonesian Food Image Dataset) [Data set]," *Kaggle*, 2022, doi: KAGGLE/DSV/4053613.
- [8] A. Wibisono, H. A. Wisesa, Z. P. Rahmadhani, P. K. Fahira, P. Mursanto, and W. Jatmiko, "Traditional food knowledge of Indonesia: a new high-quality food dataset and automatic recognition system," *Journal of Big Data*, vol. 7, no. 1, p. 69, Dec. 2020, doi: 10.1186/s40537-020-00342-5.
- [9] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.
- [10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.
- [11] Z. Liu, H. Mao, C. Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2022-June, pp. 11966–11976, Jun. 2022, doi: 10.1109/CVPR52688.2022.01167.
- [12] G. Huang, Z. Liu, and L. van der Maaten, "Densely Connected Convolutional Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 39, no. 9, pp. 1442–1446, 1978.
- [13] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.
- [14] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," *Proceedings of Machine Learning Research*, vol. 139, pp. 10096–10106, 2021.
- [15] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Jun. 2015, doi: 10.1109/CVPR.2015.7298594.
- [16] A. Howard *et al.*, "Searching for mobileNetV3," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1314–1324, Oct. 2019, doi: 10.1109/ICCV.2019.00140.
- [17] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 10425–10433, Jun. 2020, doi: 10.1109/CVPR42600.2020.01044.
- [18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Jun. 2016, doi: 10.1109/CVPR.2016.90.
- [19] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 5987–5995, Jul. 2017, doi: 10.1109/CVPR.2017.634.
- [20] N. Ma, X. Zhang, H. T. Zheng, and J. Sun, "Shufflenet V2: Practical guidelines for efficient cnn architecture design," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11218 LNCS, pp. 122–138, 2018, doi: 10.1007/978-3-030-01264-9_8.
- [21] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv*, 2016, doi: 10.48550/arXiv.1602.07360.
- [22] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9992–10002, Oct. 2021, doi: 10.1109/ICCV48922.2021.00986.
- [23] Z. Liu *et al.*, "Swin Transformer V2: Scaling Up Capacity and Resolution," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 11999–12009, Jun. 2022, doi: 10.1109/CVPR52688.2022.01170.
- [24] S. Karen and Z. Andrew, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, p. 14, 2015.
- [25] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," in *British Machine Vision Conference 2016, BMVC 2016*, pp. 87.1–87.12, 2016, doi: 10.5244/C.30.87.
- [26] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *arXiv*, 2018, doi: 10.48550/arXiv.1803.09820.





BIOGRAPHIES OF AUTHORS

Mohammad Arif Rasyidi     earned a bachelor's degree in Information Systems from Sepuluh Nopember Institute of Technology, Indonesia, in 2012 and a master's degree in Computer Engineering from Pusan National University, Korea, in 2015. His research interests include machine learning and evolutionary computation. He can be contacted at email: mohammad.rasyidi@uisi.ac.id.







Yunita Siti Mardhiyyah     earned a bachelor's degree in Food Science and Technology and a master's degree in Food Science from Bogor Agricultural University, Indonesia in 2012 and 2016 respectively. Her research interests include functional food, traditional food, food sensory analysis, and food chemistry. She can be contacted at email: yunita.mardhiyyah@uisi.ac.id.



Zuraidah Nasution     earned a bachelor's degree in Food Technology from Padjajaran University, Indonesia, in 2004, a master's degree in Food Science & Nutrition from The University of New South Wales, Australia, in 2006, and a doctorate in Food Science from Kasetsart University, Thailand, in 2018. Her research interests include food science and nutrition. She can be contacted at email: zuraidah.nasution@apps.ipb.ac.id.



Christofora Hanny Wijaya     is a professor of Flavor Chemistry and Technology. She earned a bachelor's degree in Agricultural Processing Technology from Bogor Agricultural University, Indonesia, in 1983, and both master's and doctorate in Agricultural Chemistry from Hokkaido University, Japan, in 1987 and 1990, respectively. Her research interests include Isolation and identification of flavor components and bioactive compounds from local resources, study on the volatile quality of tropical fruits produced by breeding, tropical peatland management, and Asian ethnic diets and traditional herbal medicines in the perspective of functional foods and health promotion. She can be contacted at email: hazemi@indo.net.id.